Automation beyond Testing and Embedded System Validation

Embedded Linux Conference Europe
Jan Lübbe <j.luebbe@pengutronix.de>

Pengutronix.

Some Background

- Embedded Linux integration and development for custom hardware
- Using Linux mainline, mesa, wayland, gstreamer, Qt, chromium, …

⇒ Everything changes all the time

⇒ Updates break user-visible features

Kernel and application level testing
"solved" with Jenkins & LAVA

A Short Survey

- Who has developed embedded Linux systems?
  - … rolled out a major base-system update?
  - … updates the base-system at least once a year?
- Who has automated tests for:
  - The application?
  - … and the kernel (-drivers)?
  - … and the update installer?
  - … and the rollback mechanism?
- What do you use (in-house, Jenkins, LAVA, …)?

Current State

- Test automation:
  - LAVA, Fuego, autotest, avocado, TI VATF, U-Boot test tool, CI-RT R4D, Baylibe Lab in a Box, …
- Development automation:
  - scripting via SSH
  - expect
- Production automation:
  - flash images via robot
  - SoC-vendor-sepecific tools (running on windows)
  - ad-hoc scripting

Our Wishlist

- Short turnaround times for interactive use during development
- Support reuse for other use cases
- Use the same board for devel and CI
- Complex state transitions (BL → Linux → update & reboot → BL → new Linux)
- Library interface (for use-cases besides testing)
- Control of additional interfaces (SD, buttons, boot mode, logic analyzer, USB, …)
  - supported by TI's VATF for some special cases
- Multiple targets in one test case
  - supported by LAVA

NIH Syndrome?

- All tools are shaped by requirements
- Our use-cases != your use-cases

LAVA - Linaro Automated Validation Architecture

- Used by Linaro, Kernel CI and many others
- lavapdu daemon

✔good web-interface with useful logs
✔automatic health checks

✖ boards must be dedicated to LAVA
✖ long turn-around times

see “Introducing the Lab in a Box Concept” tomorrow (http://sched.co/ByYM)

Pengutronix.

Fuego

- Used by LF CE WG, LTSI (Long Term Support Initiative), AGL, CIP
- Consists of Jenkins + Scripts + Tests (in Docker)

✖ builds test and deploys test binaries
✖ hard to setup on an existing Jenkins instance

U-Boot “pytest suite”

- Lives in u-boot/test/py
- Helper functions to build and control U-Boot

✔expressive test cases using pytest

✖ only for U-Boot (with build support)
✖ no library interface or target abstraction

CI-RT R4D

- Power & serial control
- Implemented as libvirt backend

✔embedded boards controlled similar to VMs
✔easy to use from Jenkins

✖ libvirt interface does not fit more complex use cases
✖ difficult to synchronize multi node tests
✖ needs custom code fo rinterfaces besides power and serial

see “CI: Jenkins, libvirt and Real Hardware” (http://sched.co/ByYA)

Heiko Schocher's tbot

- Python tool to control boards and execute test cases

✔ access to remote boards via SSH
✔ flexible event collection for reporting

✖ patch and build support
✖ plain python code for testcases (instead of pytest)

Project Specific Tools

- Autotest fork by Google for Chromium OS
- Avocado (another Autotest fork) for libvirt testing
- TI's VATF

✖ directly contain the testsuites
✖ very focused on special requirements
✖ only for testing

Shortcomings

- Large overhead for running and writing a single test
  ⇒ painful to use during iterative development
- Limited control over the target from the individual test
  - no reboots during test case
  - no easy control over additional IO (buttons, config switches, USB, …)
- Hard to reuse for other use-cases and one-off tools
  - git bisect
  - …

Goals

- Make automation useful during normal iterative development
  - Upload bootloader via USB
  - Control distributed equipment
  - Easy test loops
- Support the same tests and tools from a CI environment
- Make it easy to extend and embed
- Connect/automate existing tools (LTP, …)

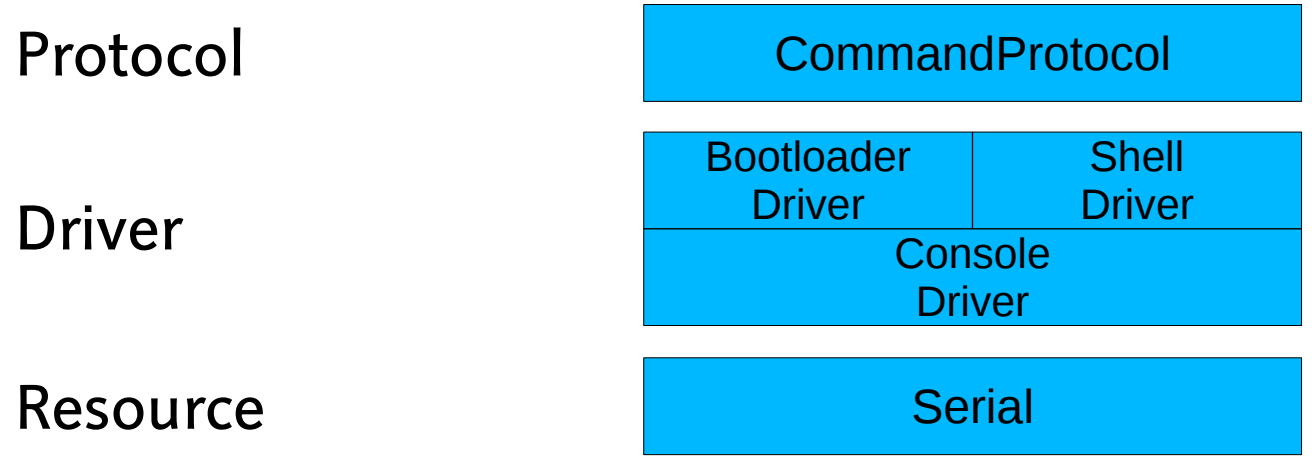
Try ~~Something Else~~ Less

- no integrated build system (unlike Fuego)
  - use OE/PTXdist/buildroot instead
- no integrated test runner (unlike LAVA, autotest, many others)
  - use pytest and/or custom scripts
- no scheduler (unlike LAVA, Fuego)
  - use Jenkins instead or use from shell
- no fixed boot process (all? others)
  - full control from client code
- library interface ⇒ not only for testing

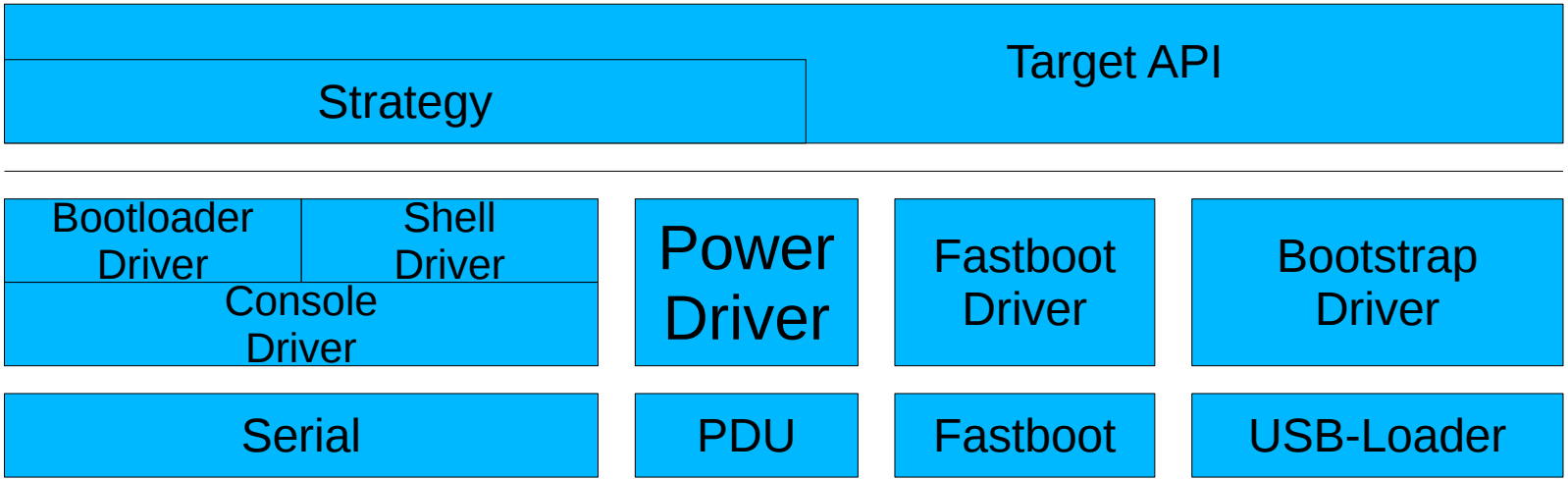
HW/SW Control as a Library

- Embedded system testing should feel like pure SW testing
- Don’t handle control-flow
- Client code should be high-level
  - Similar to what I would tell a colleague to do

Labgrid - Architecture

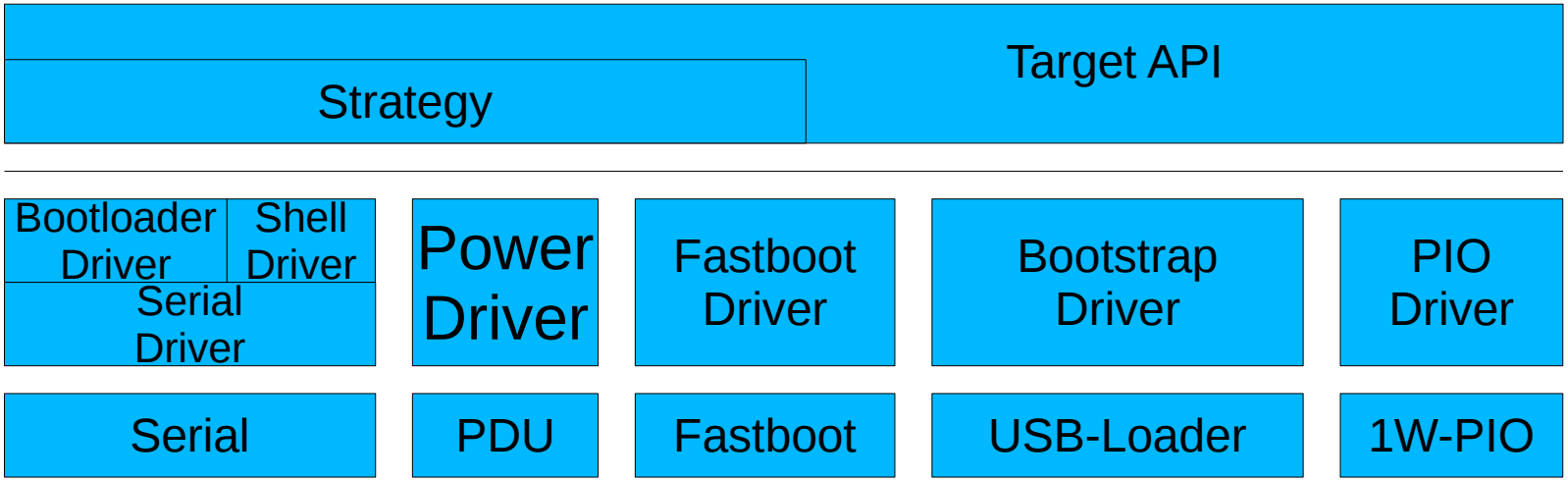


Architecture – Targets



Pengutronix.

Architecture - Flexibility



Pengutronix.

Labgrid - Configuration

- YAML
- Describes Targets with
  - Resources
  - Drivers
- HW/SW-Specific parameters
- The “Environment”

```
targets:
  main:
    resources:
      RawSerialPort:
        port: "/dev/ttyUSB0"
    drivers:
      ManualPowerDriver:
        name: "example"
      SerialDriver: {}
      BareboxDriver:
        prompt: 'barebox@[^:]+:[^ ]+ '
      ShellDriver:
        prompt: 'root@\w+:[^ ]+ '
        login_prompt: ' login: '
        username: 'root'
      BareboxStrategy: {}
```

Labgrid - pytest

- Test execution, selection and reporting is provided by pytest
- Fixtures provide access at different levels (command, strategy, target, env)
- pytest (and Python libs) make it easy to prepare test data and analyze results
- Easy to integrate in Jenkins

```
def test_hwclock_rate(command):
    """Test that the hardware clock rate is not too inaccurate."""
    result = command.run_check('hwclock -c | head -n 3')
    hw_time, sys_time, freq_offset_ppm, tick = result[-1].strip().split()
    assert abs(int(freq_offset_ppm)) < 1000
```

Test Result

9 failures (+4) , 15 skipped (±0)

207 tests (+6)

Took 10 min.

add description

All Failed Tests

Test Name	Duration	Age
tests.test_userspace_services.test_wifi_regulatory_domain	2 sec	1
tests.test_userspace_services.test_ubihealthd	3.1 sec	1
tests.test_userspace_services.test_barebox_healthd	2.1 sec	1
tests.test_userspace_services.test_rfkill	16 sec	1
tests.test_linux_interfaces.test_network_interfaces	2 sec	2
tests.test_linux_interfaces.test_bluetooth_interfaces	1.9 sec	2
tests.test_linux_interfaces.test_loaded_modules	2 sec	2
tests.test_linux_ecryptfs.test_linux_ecryptfs_dep	4.1 sec	35
tests.test_linux_nvmem.test_linux_nvmem_nvstore	20 sec	108

All Tests

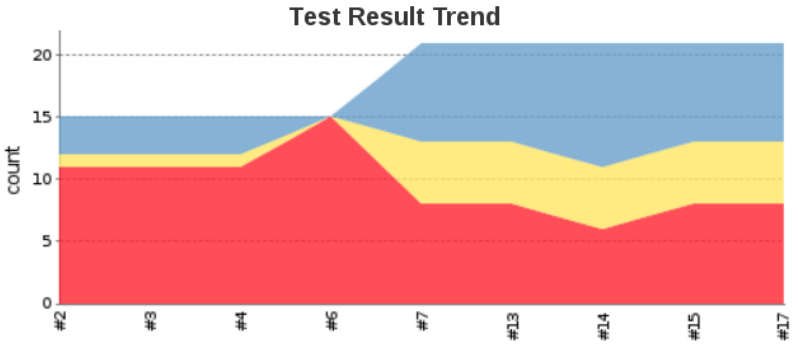
Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
tests	10 min	9	+4	15		183	+2	207	+6

Pipeline jlu/rauc/status-file

Full project name: integration-tests/combined/jlu%2Frauc%2Fstatus-file

Recent Changes

Stage View



(just show failures) enlarge

	Declarative: Checkout SCM	SCM	Build ptxdist	Prepare	Build BSP	Test (pytest-barebox)	Test (pytest-shell)	Test (pytest-rauc)	Test (reason)
Average stage times:	18s	4min 42s	20s	24s	17min 23s	6s	43s	59s	45s
#17 Oct 17 17:37 — 1 commits	2s	11s	18s	32s	23min 49s	6s	1min 38s	2min 12s	2min 4s
#16 Oct 17 12:07 — 5 commits	31s	1min 15s	33s	37s failed	310ms	81ms	70ms	315ms	207ms
#15 Oct 13 — 1	981ms	34s	11s	15s	38min 4s	7s	1min 23s	2min 5s	1min 36s

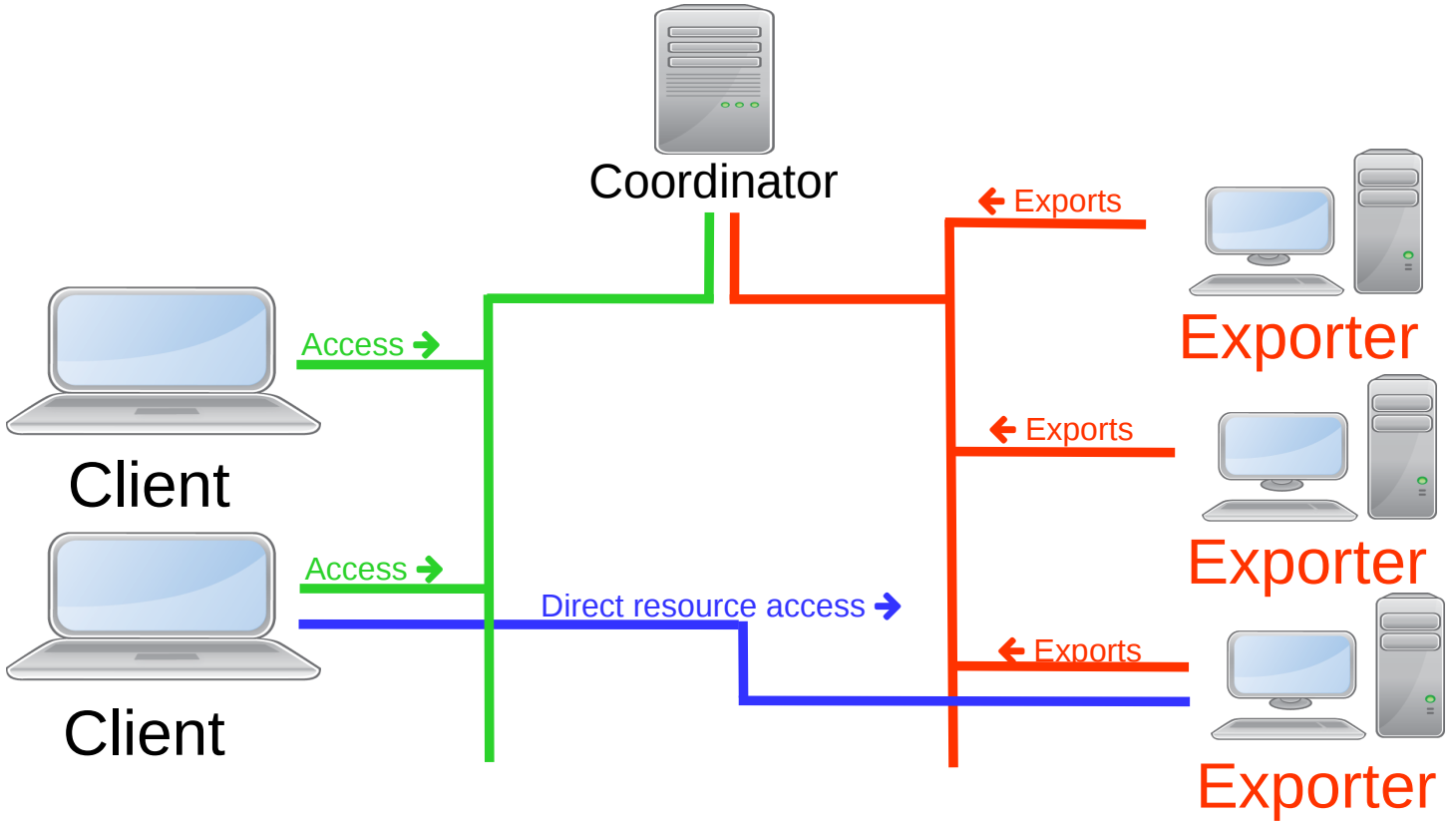
Pengutronix.

Labgrid - CLI

- Configure boards from distributed resoures (“board farms”/“labs”)
- Control power, serial, buttons, fastboot, bootloader upload
- Lock/unlock boards
- Use labgrid strategies
- Usable from shells scripts, CI or other automation (such as LAVA)

```
labgrid-client -p riot1 lock
labgrid-client -p riot1 io high bootmode
labgrid-client -p riot1 pw on
labgrid-client -p riot1 bootstrap …/bootload.img
labgrid-client -p riot1 fastboot boot …/kernel.img
labgrid-cleint -p riot1 console
```

Labgrid – Remote Control



Pengutronix.

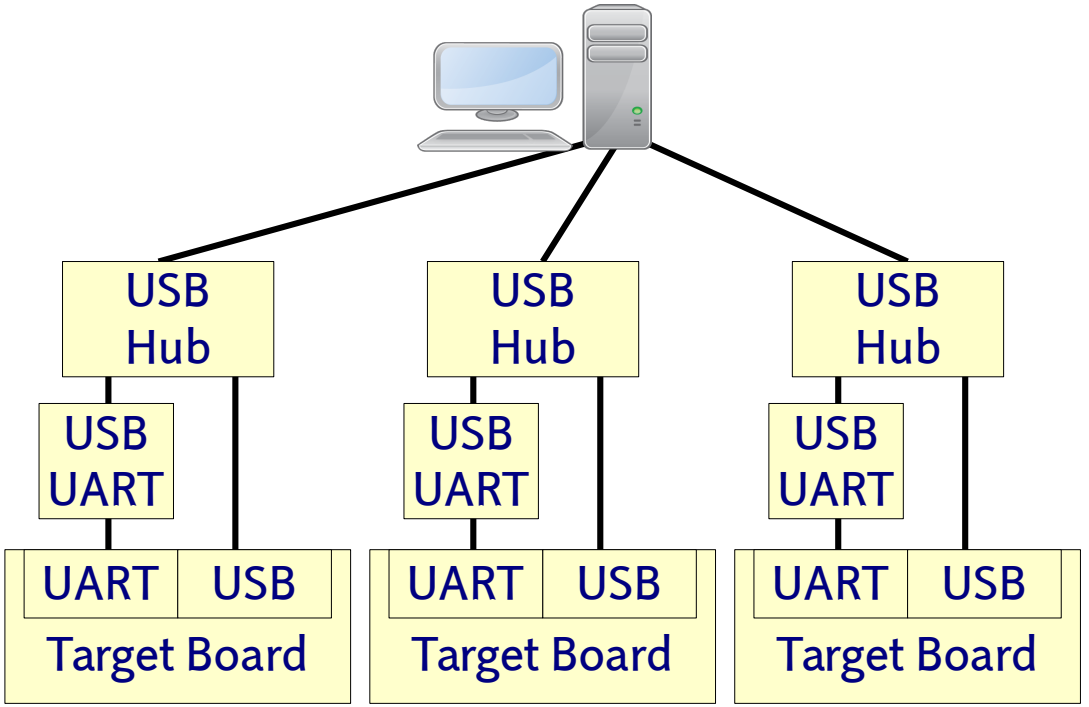
Labgrid - Scripting

- Example: Sometimes a ethernet interface discards frames instead of sending them in ~1% of boots.
- Loop until error occours
- Manual investigation after script exits

```
def check_port(eth):
    command = target[ShellDriver]
    _, _, _ = command.run('arping -I {} 1.2.3.4 -c1'.format(eth))
    stdout, _, ret = command.run('ethtool -S {}'.format(eth))
    if ret:
        return False
    for line in stdout:
        ... parsing ...
        if k == 'good_frames_sent':
            if int(v) == 0:
                return False
    return True

def run_test(target):
    strategy = target[MyBareboxStrategy]
    strategy.transition('off')
    strategy.transition('shell')

    if not check_port('eth0'):
        return False
    if not check_port('eth2'):
        return False
    return True

env = Environment('myboard.yaml')
target = env.get_target()
for i in range(1000):
    if not run_test(target):
        break
```

Labgrid - Autoinstaller

- Each host manages serveral flashing stations
- Uses USB tree topology for configuration



https://github.com/labgrid-project/labgrid/blob/master/labgrid/autoinstall/main.py

Pengutronix.

Demo

Currently Working

- Remotly control boards in lab from CLI (console, power, BL upload, fastboot)
- Run pytest against local and remote boards
- Run tests from Jenkins and collect results via Junit-XML
- Ad-Hoc automation: git bisect, reproducing sporadic errors
- Automatic factory installation via USB directly from built BSPw
- Used as a backend for internal QA tools

Next Steps

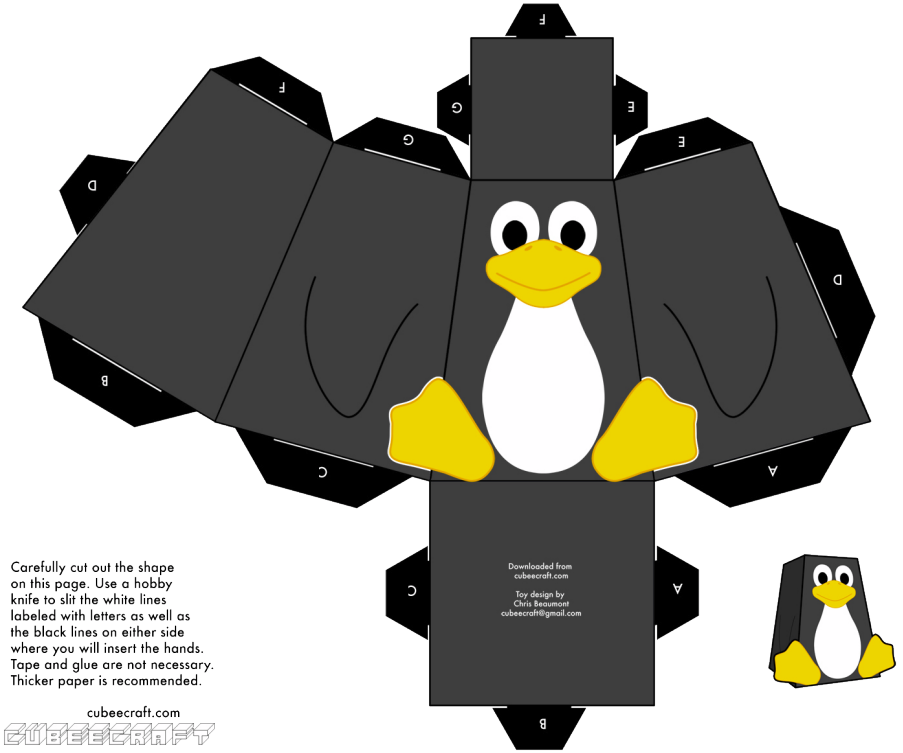
- Remote target reservation (for use with Jenkins CI)
- Automatic integration tests for RAUC with QEmu in Jenkins
- Improved logging and reports
- Driver priorities (use ResetProtocol instead of PowerProtocol when available)
- Driver preemption (handle unexpected state changes)

Getting Started

- Wait for 0.2.0 release or git clone master
- Setup in Python venv
- Connect board
- Copy and modify one of the examples
- If it breaks: talk to us! ;-)

https://labgrid.readthedocs.io/en/latest/getting_started.html

Discussion



jlu@pengutronix.de, @shoragan, +JanLübbe-jlu

Pengutronix.